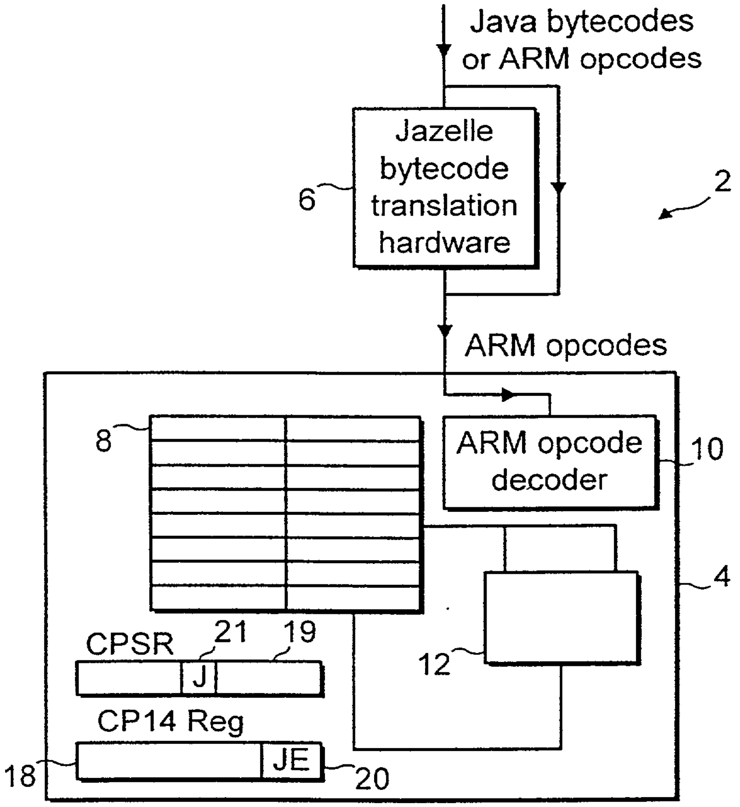
Java bytecodes
or ARM opcodes
Jazelle
bytecode
translation
hardware
6
2
ARM opcodes
8
ARM opcode
decoder
10
4
12
CPSR
21
19
J
CP14 Reg
JE
18
20

FIG. 1

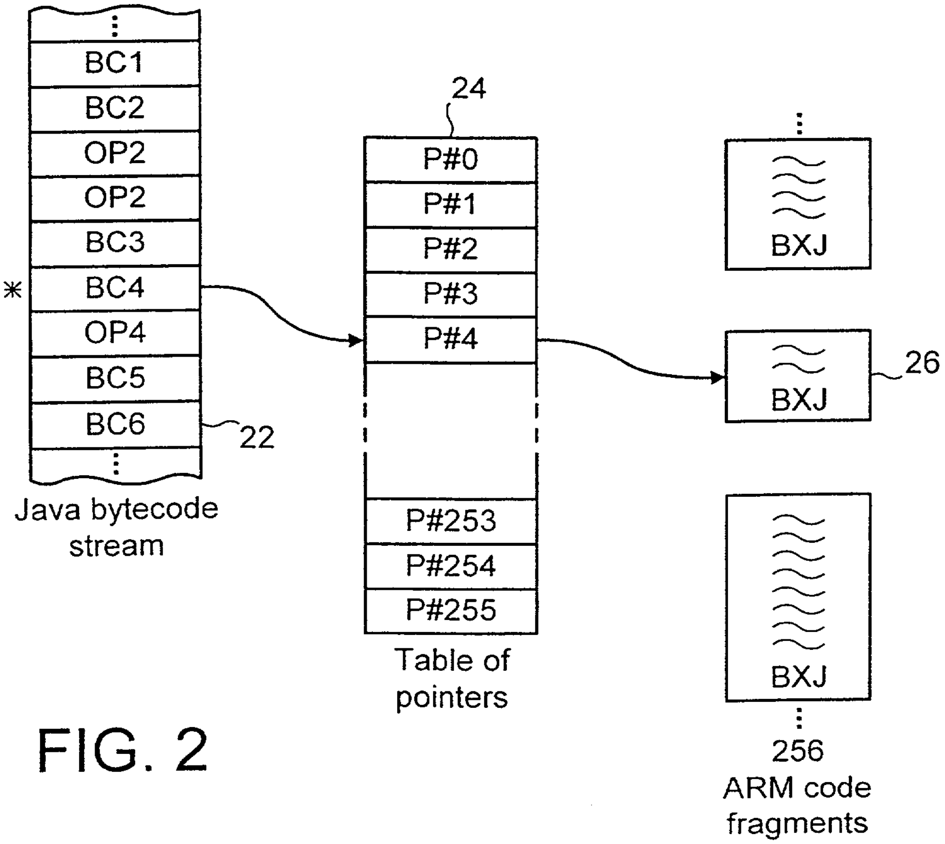
BC1
BC2
OP2
OP2
BC3
*
BC4
OP4
BC5
BC6
22
Java bytecode
stream
24
P#0
P#1
P#2
P#3
P#4
P#253
P#254
P#255
Table of
pointers
BXJ
BXJ
26
BXJ
256
ARM code
fragments

FIG. 2

10031215.022502

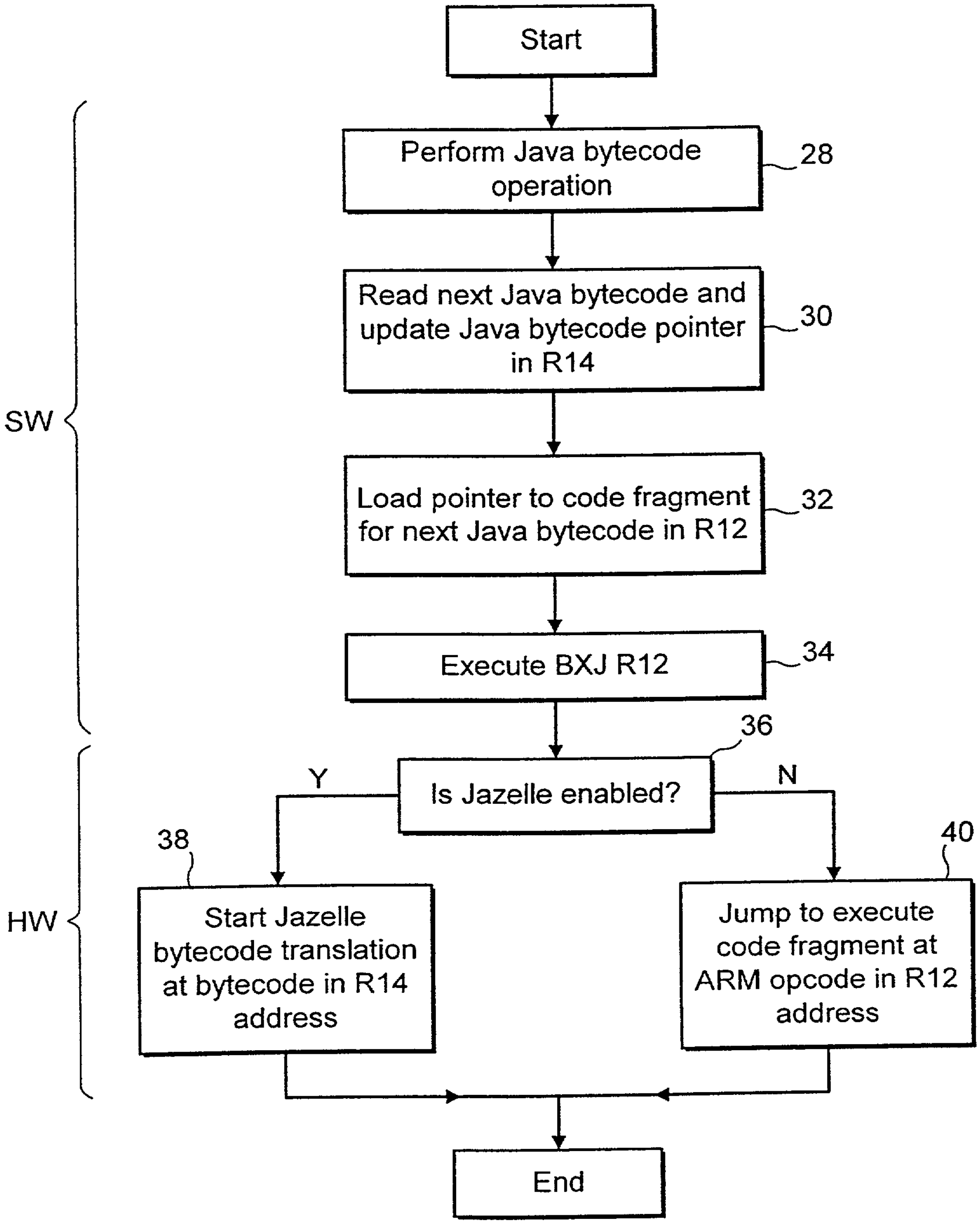
Start
Perform Java bytecode operation
28
Read next Java bytecode and update Java bytecode pointer in R14
30
Load pointer to code fragment for next Java bytecode in R12
32
Execute BXJ R12
34
SW
36
Is Jazelle enabled?
Y
N
38
Start Jazelle bytecode translation at bytecode in R14 address
40
Jump to execute code fragment at ARM opcode in R12 address
HW
End

FIG. 3

```
do_iadd
LDRB    R4, [R14, #1]! ———————— Load next Java bytecode and update bytecode pointer
LDR     R1, [Rstack, #-4]! ————— POP first operand from stack
LDR     R0, [Rstack, #-4]! ————— POP second operand from stack
LDR     R12, [Rexc, R4, LSL #2] —— Get address of code fragment for next bytecode
ADD     R0, R0, R1 ———————————— Perform integer add
STR     R0, [Rstack], #4 —————— PUSH result to stack
BXJ     R12 ——————————————————— Do next bytecode in hardware/software
```

FIG. 4

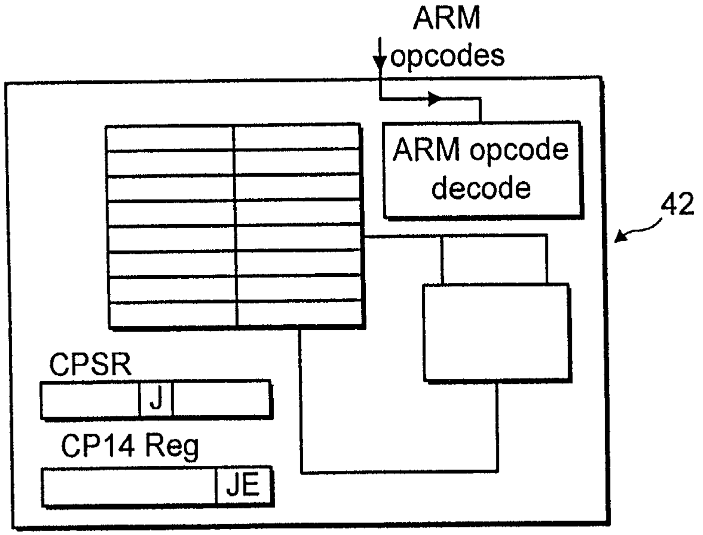

FIG. 5

10081215 . 022502

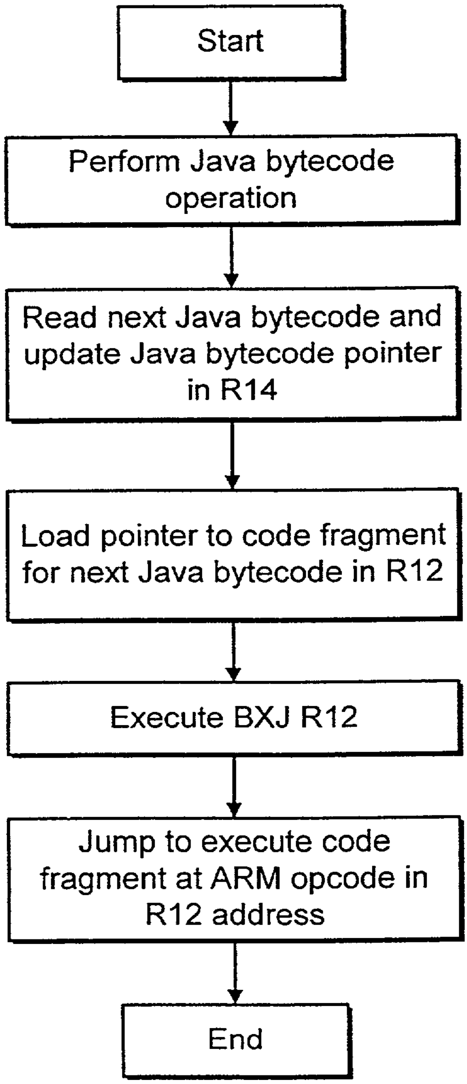
Start
Perform Java bytecode operation
Read next Java bytecode and update Java bytecode pointer in R14
Load pointer to code fragment for next Java bytecode in R12
Execute BXJ R12
Jump to execute code fragment at ARM opcode in R12 address
End

FIG. 6

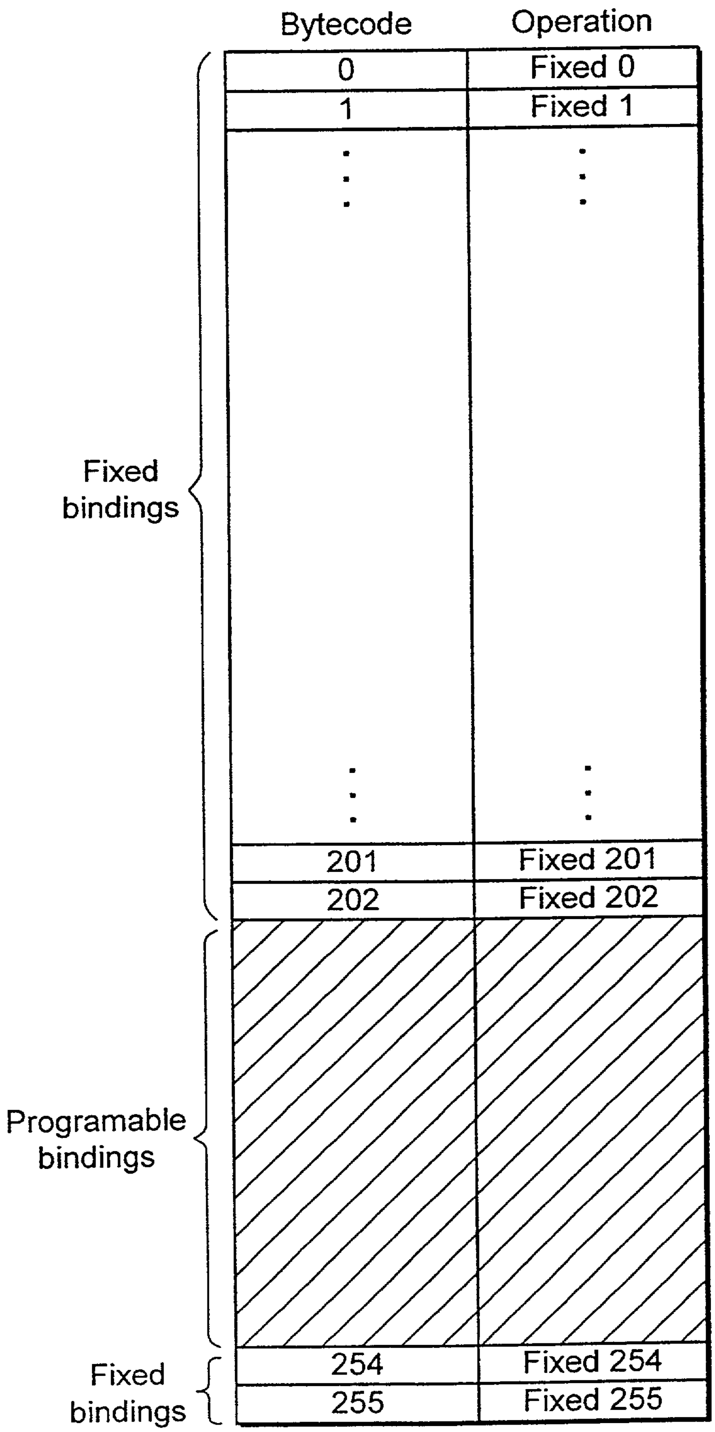
Bytecode
Operation
0
Fixed 0
1
Fixed 1
201
Fixed 201
202
Fixed 202
254
Fixed 254
255
Fixed 255
Fixed bindings
Programable bindings
Fixed bindings

FIG. 7

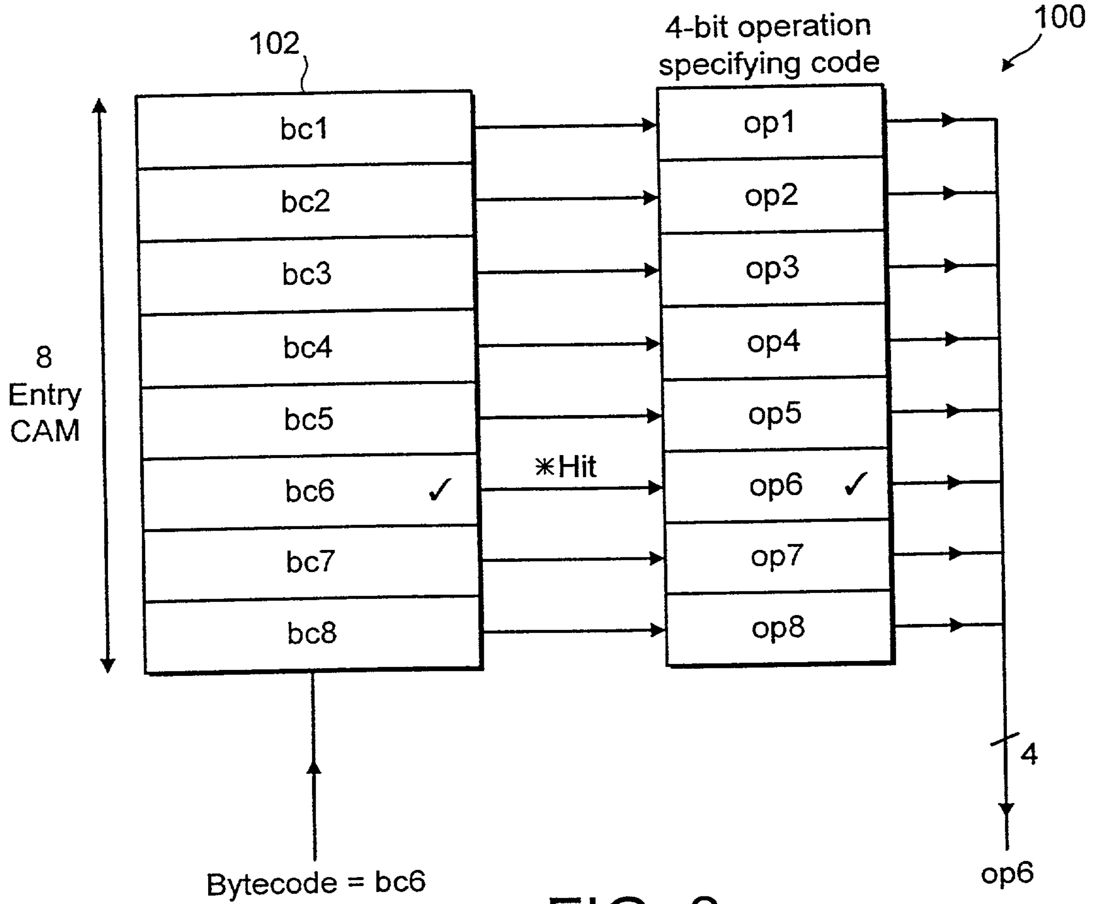
102
4-bit operation specifying code
100
8 Entry CAM
bc1
bc2
bc3
bc4
bc5
bc6 ✓
bc7
bc8
*Hit
op1
op2
op3
op4
op5
op6 ✓
op7
op8
4
Bytecode = bc6
op6

FIG. 8

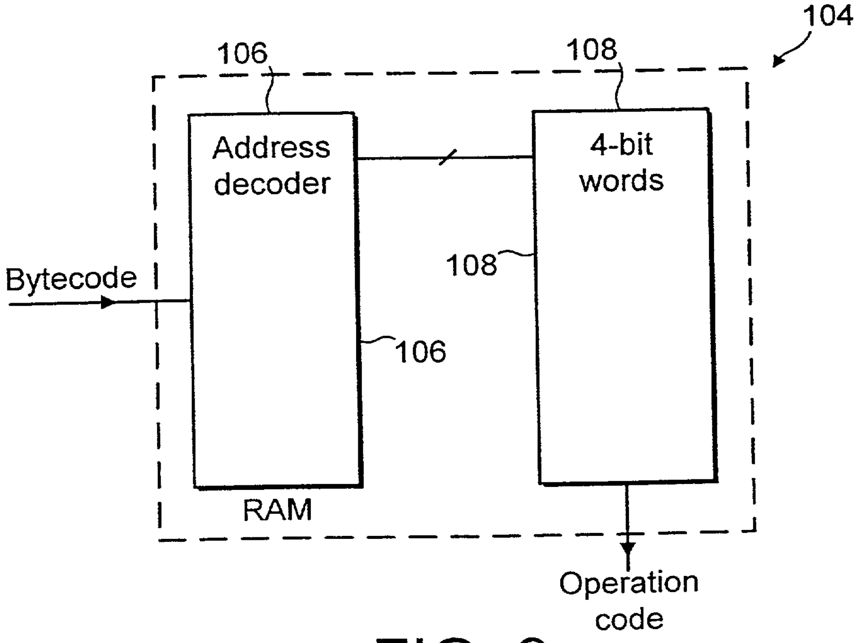
104
106
108
Address decoder
4-bit words
108
Bytecode
106
RAM
Operation code

FIG. 9

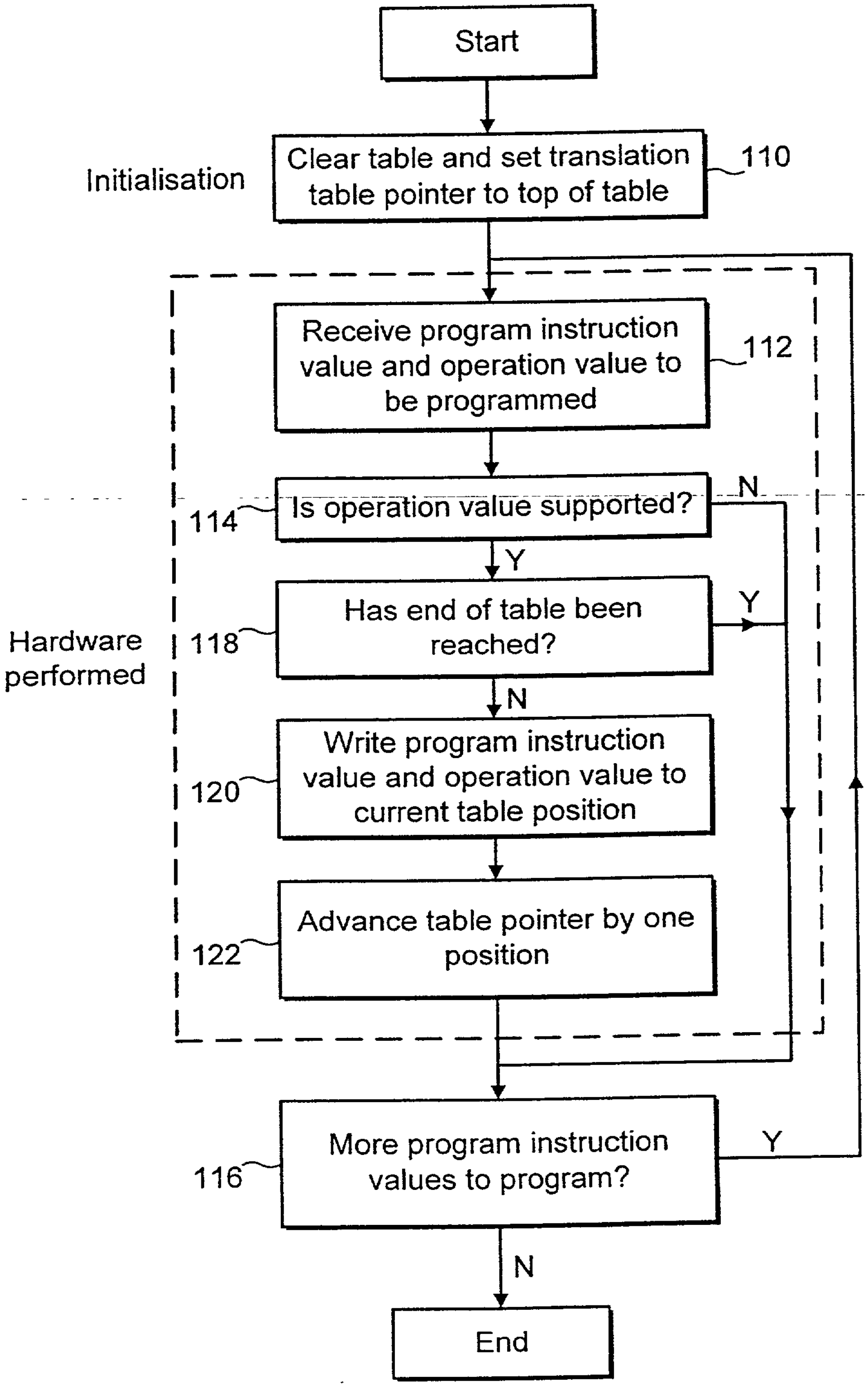
Start
Initialisation
Clear table and set translation table pointer to top of table
110
Receive program instruction value and operation value to be programmed
112
Is operation value supported?
114
N
Y
Hardware performed
Has end of table been reached?
118
Y
N
Write program instruction value and operation value to current table position
120
Advance table pointer by one position
122
More program instruction values to program?
116
Y
N
End

FIG. 10

10052525 - 092508

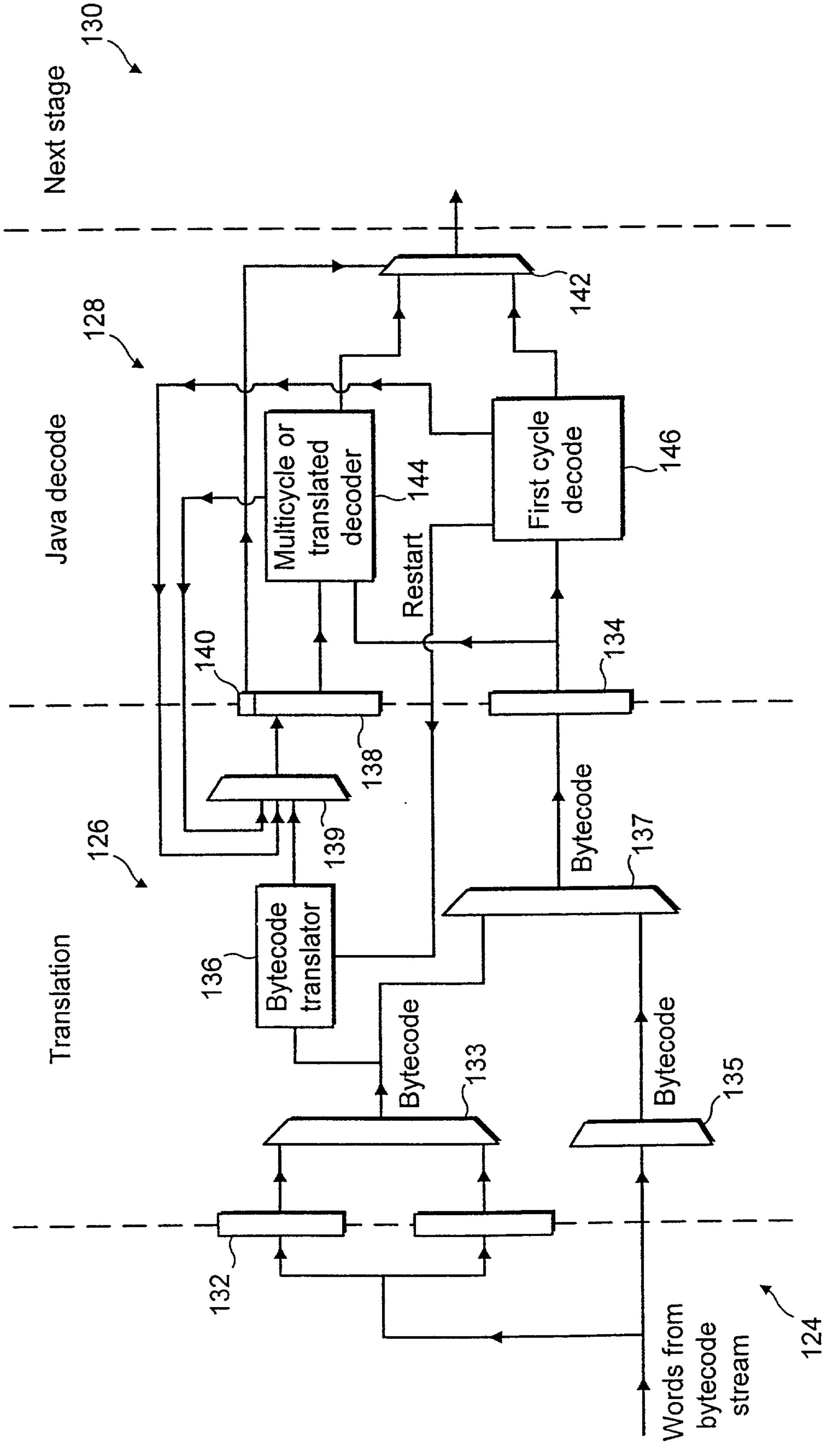
Translation
Java decode
Next stage
130
128
126
124
Words from bytecode stream
132
133
135
Bytecode
Bytecode
136
Bytecode translator
137
Bytecode
139
138
140
134
Restart
Multicycle or translated decoder
144
First cycle decode
146
142

FIG. 11

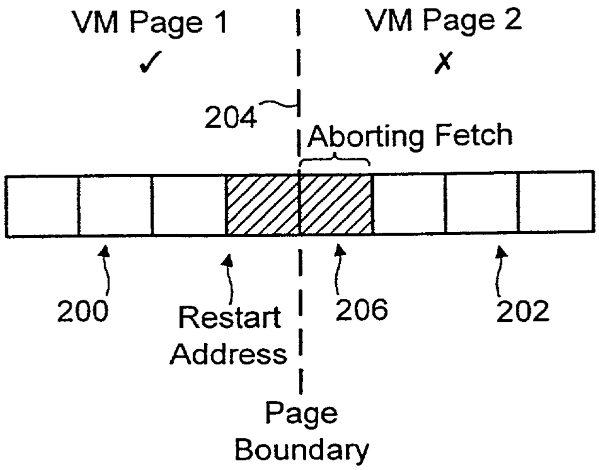

FIG. 12

(PA (Half1) = False) AND (PA (Half2) = True)

AND

(((Number of operands = 1) AND (bcadd [1:0] = 11))
OR ((Number of operands = 2) AND (bcadd [1] = 1)))

FIG. 14

10081215.022502

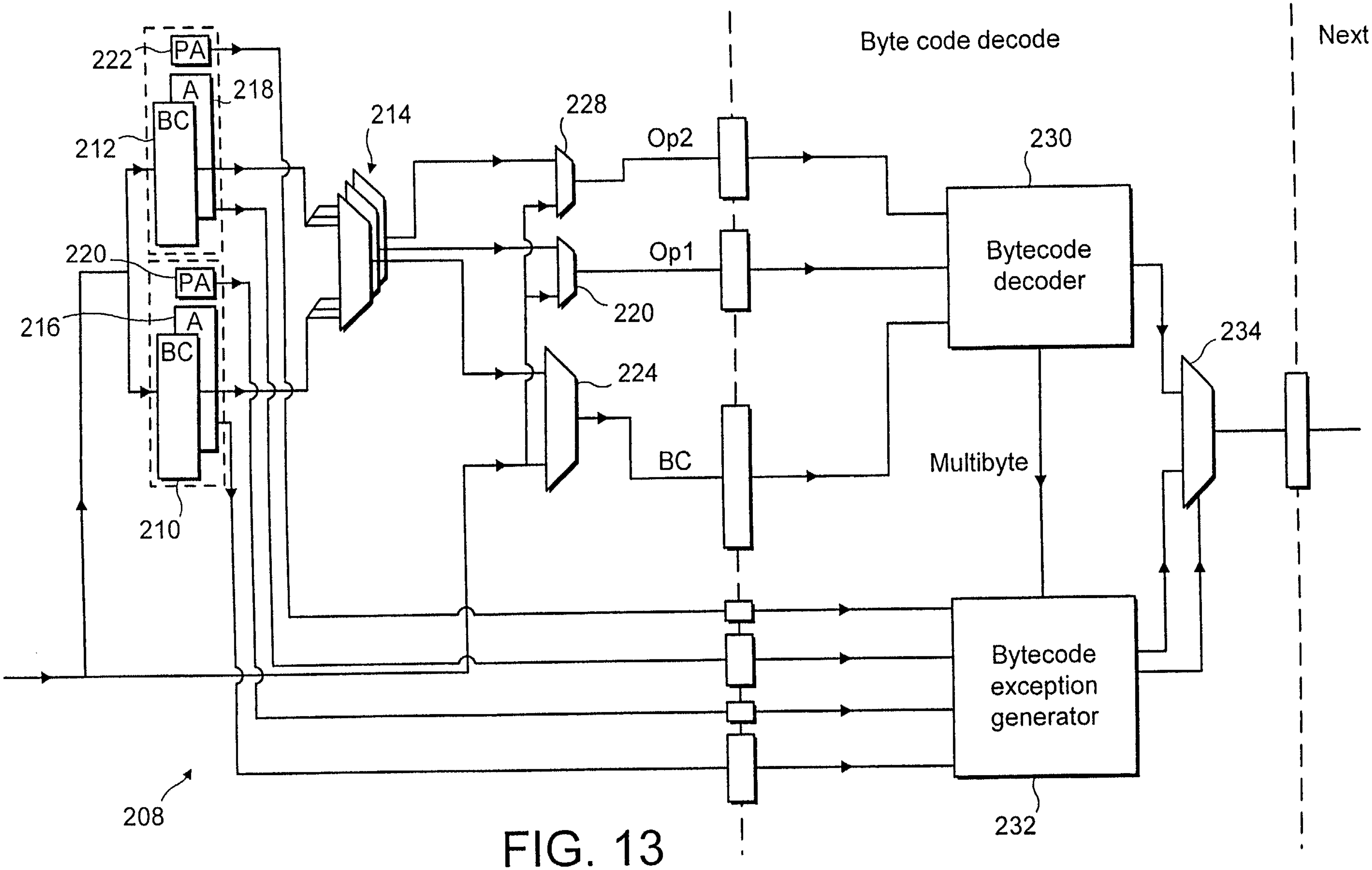
Byte code decode
Next stage
222
PA
A
218
BC
212
214
228
Op2
230
Bytecode decoder
220
PA
216
A
BC
Op1
220
234
224
BC
Multibyte
210
Bytecode exception generator
208
232

FIG. 13

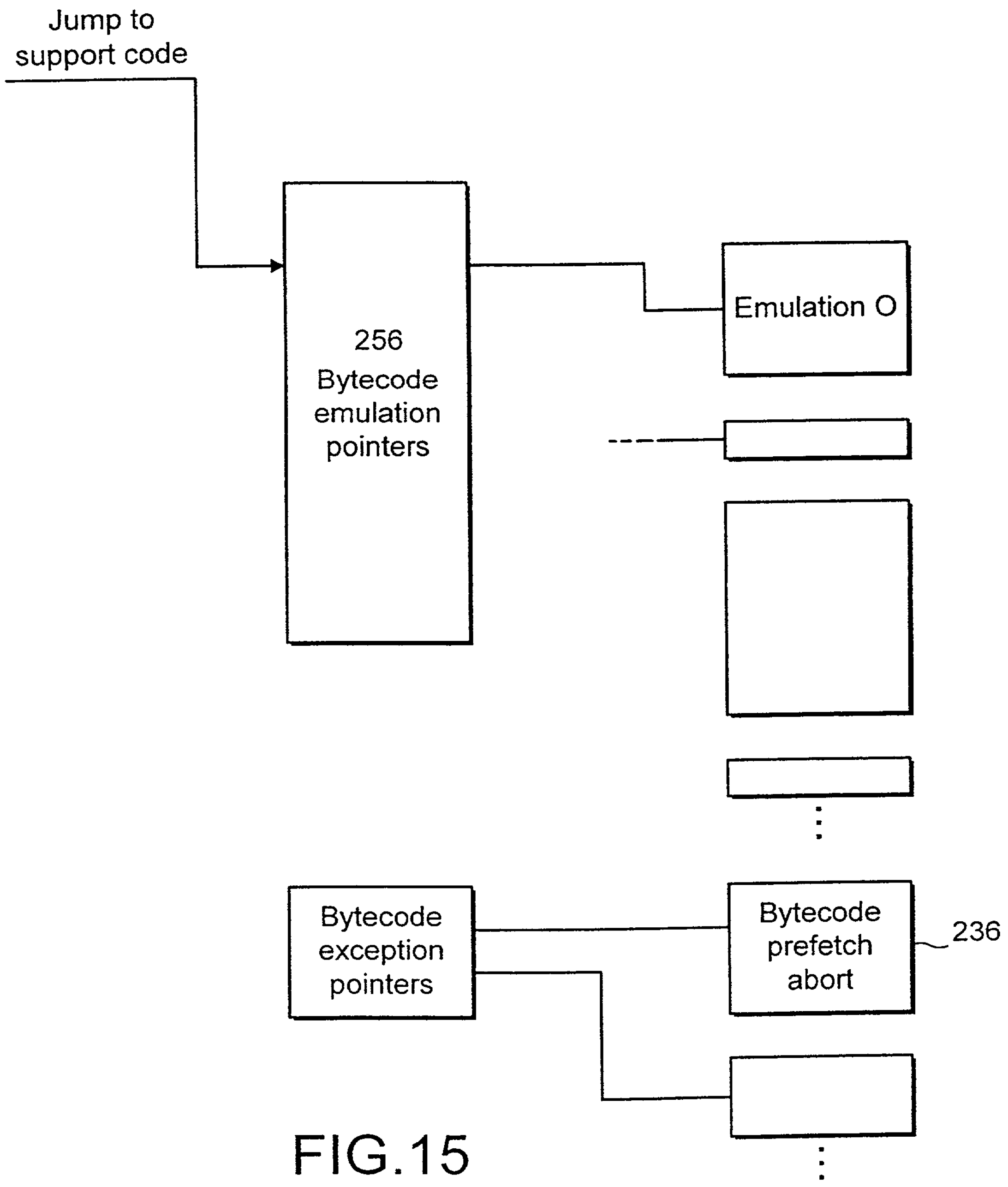
Jump to support code
256
Bytecode emulation pointers
Emulation O
Bytecode exception pointers
Bytecode prefetch abort
236

FIG.15

10051215.022502

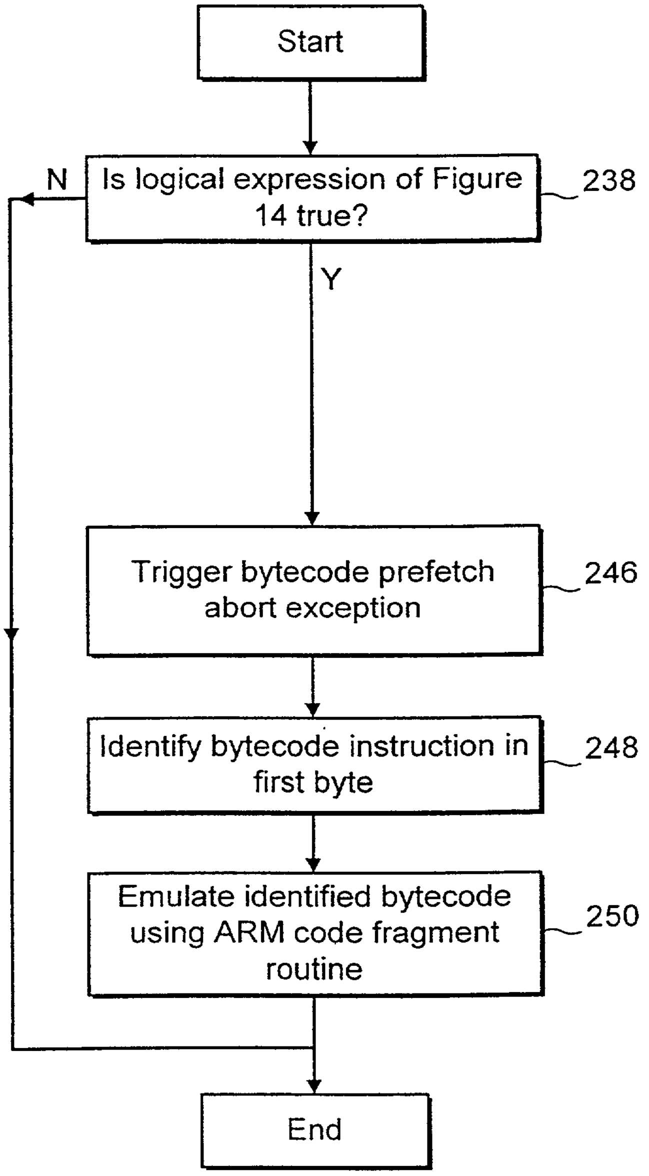
Start
Is logical expression of Figure 14 true?
238
N
Y
Trigger bytecode prefetch abort exception
246
Identify bytecode instruction in first byte
248
Emulate identified bytecode using ARM code fragment routine
250
End

FIG. 16

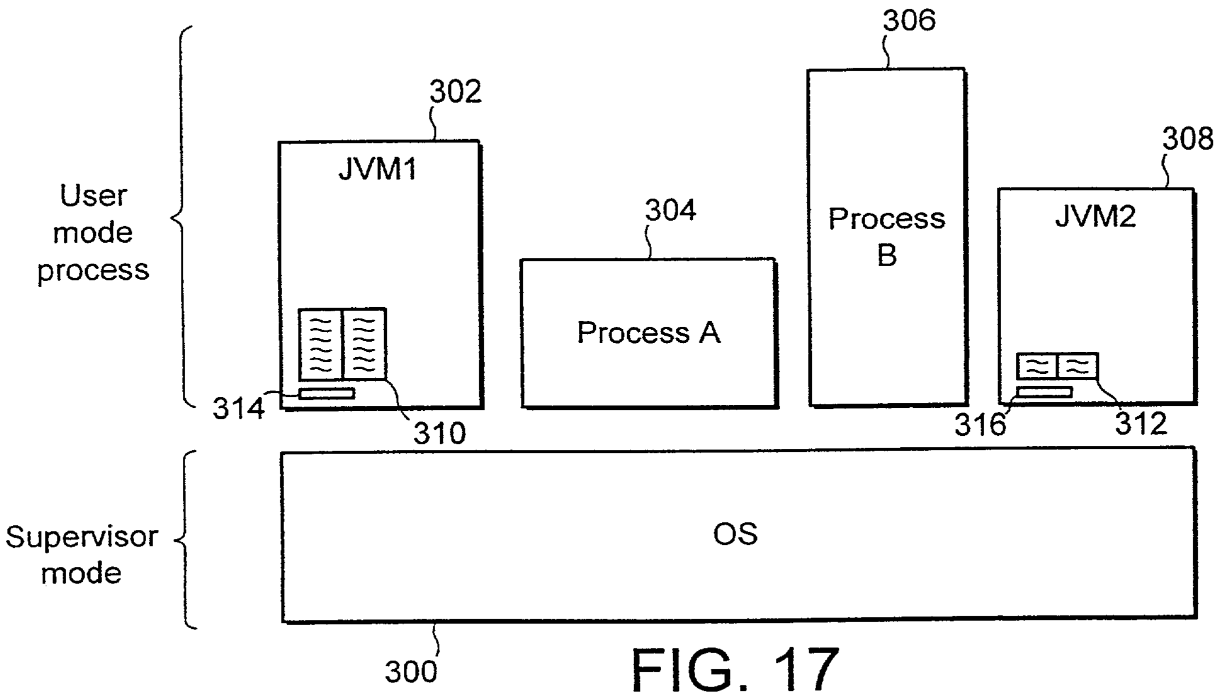
306
302
308
JVM1
304
Process B
JVM2
User mode process
Process A
314
310
316
312
Supervisor mode
OS
300

FIG. 17

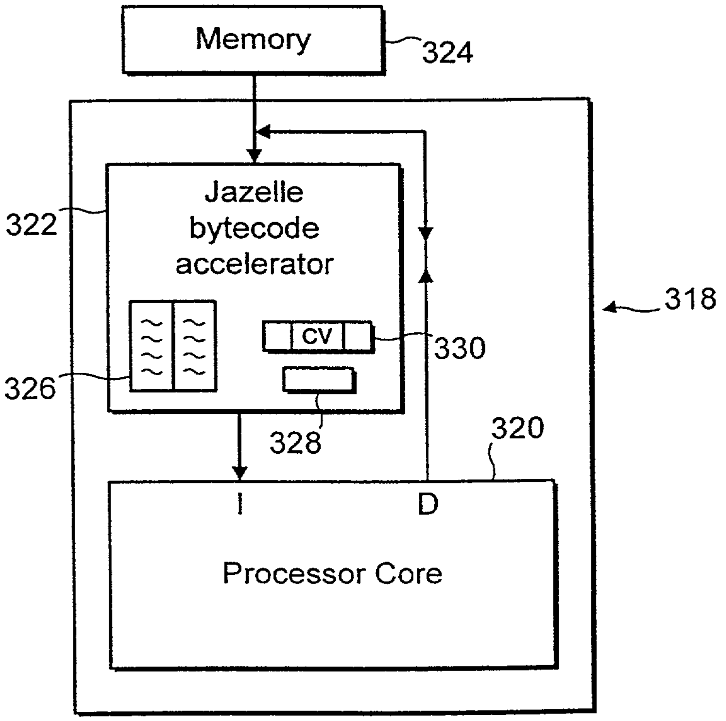
Memory
324
322
Jazelle bytecode accelerator
318
CV
330
326
328
320
I
D
Processor Core

FIG. 18

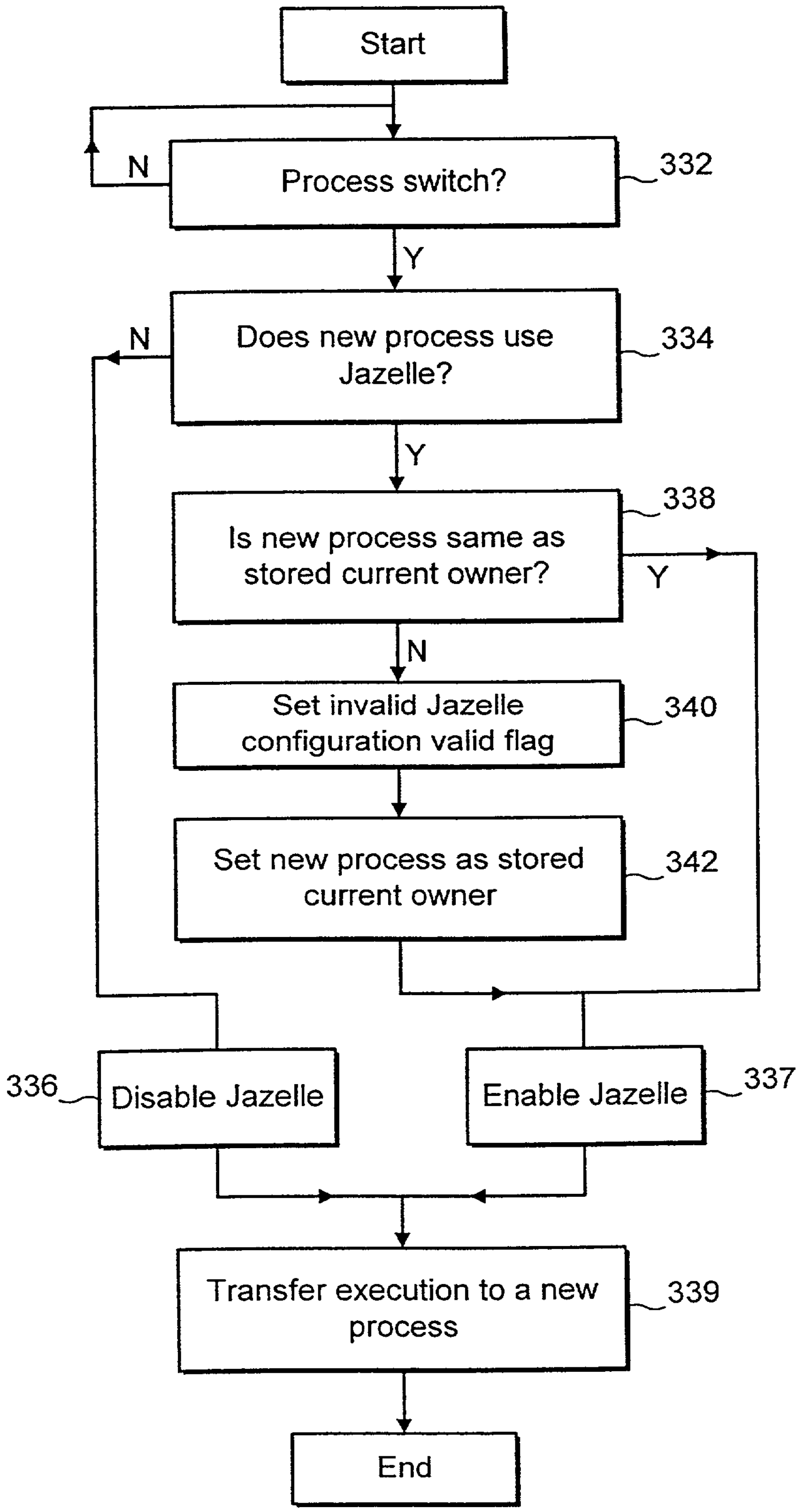
Start
Process switch?
N
Y
332
Does new process use Jazelle?
N
Y
334
Is new process same as stored current owner?
Y
N
338
Set invalid Jazelle configuration valid flag
340
Set new process as stored current owner
342
Disable Jazelle
336
Enable Jazelle
337
Transfer execution to a new process
339
End

FIG. 19

20520 - 51215060

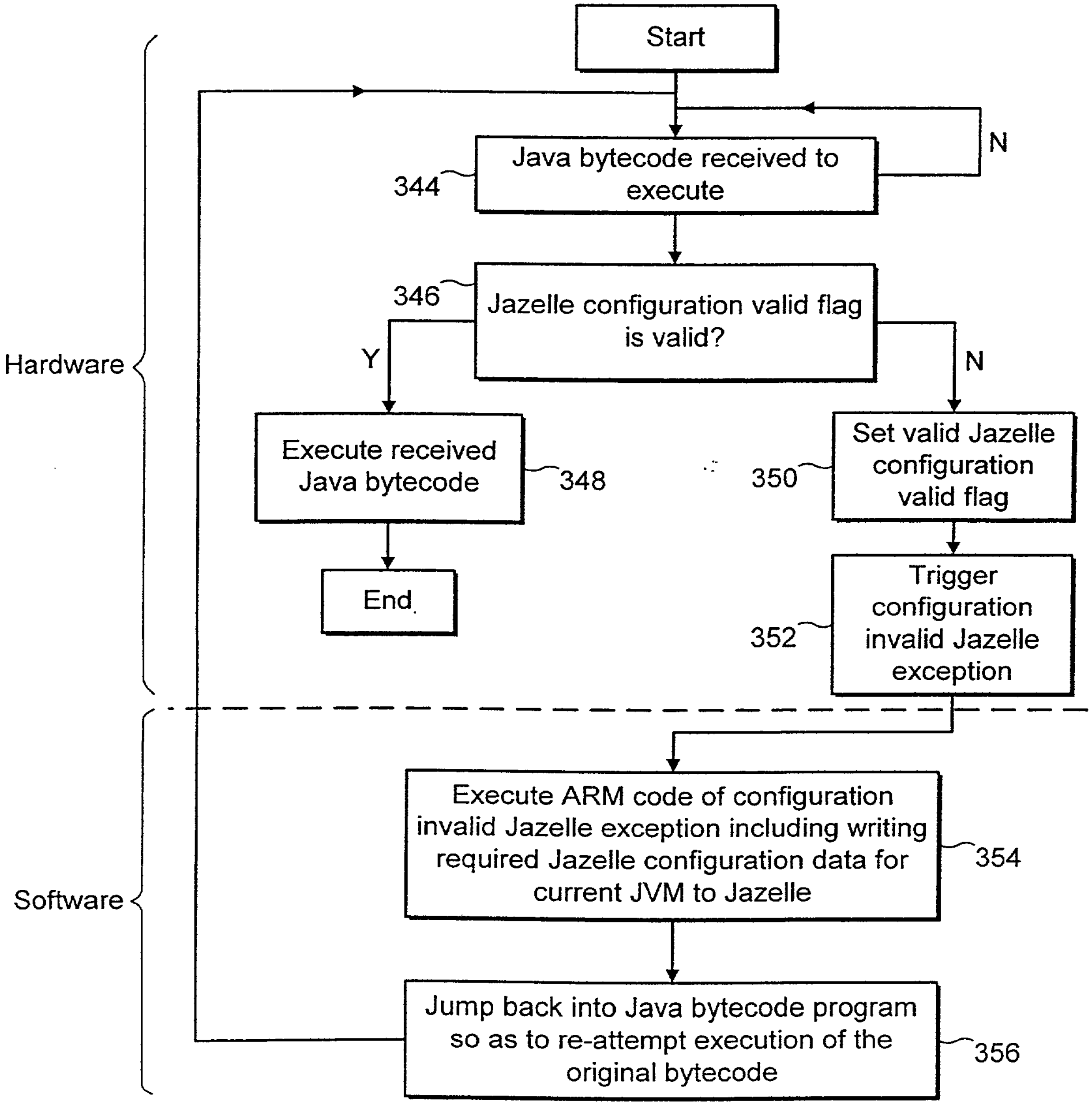
Start
Java bytecode received to execute
344
N
346
Jazelle configuration valid flag is valid?
Y
N
Execute received Java bytecode
348
End
350
Set valid Jazelle configuration valid flag
352
Trigger configuration invalid Jazelle exception
Hardware
Software
Execute ARM code of configuration invalid Jazelle exception including writing required Jazelle configuration data for current JVM to Jazelle
354
Jump back into Java bytecode program so as to re-attempt execution of the original bytecode
356

FIG. 20

10091215.022502

```
+----------------+
|                |   +------------------+
| Figure 1       |-----| Floating Point   |
|                |   | Subsystem        |
|                |   |                  |
|                |   +------------------+
+----------------+
```

FIG. 21

```
+----------------+
|                |   +------------------+
| Figure 1       |-----| Floating Point   |
|                |   | Subsystem        |
|                |   |                  |
|                |   | +-----------+    |
|                |   | |           | <------ Floating Point Operation Register
|                |   | +-----------+    |
|                |   | +-+              |
|                |-----| | | <--------------- Unhandled Operation State Flag
|                |   | +-+              |
|                |   +------------------+
+----------------+
```

FIG. 22

Single precision		Double precision	
fadd	FADDS Sd, Sn, Sm	dadd	FADDD Dd, Dn, Dm
fsub	FSUBS Sd, Sn, Sm	dsub	FSUBD Dd, Dn, Dm
fmul	FMULS Sd, Sn, Sm	dmul	FMULD Dd, Dn, Dm
fdiv	FDIVS Sd, Sn, Sm	ddiv	FDIVD Dd, Dn, Dm
frem	Not implemented in HW	drem	Not implemented in HW
fneg	FNEGS Sd, Sm	dneg	FNEGD Dd, Dm
f2d	FCVTDS Dd, Sm	d2f	FCVTSD Sd, Dm
f2i	FTOSIZS Sd, Sm	d2i	FTOSIZD Sd, Dm
f2l	Not implemented in HW	d2l	Not implemented in HW
i2f	FSITOS Sd, Sm	i2d	FSITOD Dd, Sm
l2f	Not implemented in HW	l2d	Not implemented in HW
fcmpl	FCMPS/FMSTAT	dcmpl	FCMPD/FMSTAT
fcmpg	FCMPS/FMSTAT	dcmpg	FCMPD/FMSTAT

FIG. 23

```
dmul
        FMULD   D1, D2, D1
dcmpg
        FCMPD   D0, D1
        FMSTAT
        MVNMI   R0, #0
        MOVEQ   R0, #0
        MOVGT   R0, #1
<next Java bytecode>
```

FIG. 24

```
FMULD   D1, D2, D1               ; Causes unhandled operation
FCMPD   D0, D1                   ; Unhandled operation signalled here
        |
        +---------------------+
         To ARM state to      |         ; unhandled operation handler executes
         emulate byte code    |         ; VFP read status instruction to
                    FMRX Rd, FPSCR      ; trigger VFP exception handling
                              |         ; if unhandled operation state flag
                              |         ; is set
                              +-------------------------+
            Unhandled operation state flag is           |
            set so perform VFP exception handling       |
                                                        |
                         <Read Floating Point Operation Register>
                         <Emulate FMULD tos-1, tos, tos-1 in SW>
                         <Clear Unhandled Operation State Flag>
                                                        |
                              +-------------------------+
                              | Return to unhandled operation handler
           <Re-attempt execution of FMRX Rd, FPSCR instruction,
           without triggering VFP exception handling this time>

           <Flush Java stack to memory>

           LDRB     R4, [R14]                ; Load bytecode which triggered
                                             ; unhandled operation
           LDR      R12, [Rexc, R4, LSL #2]  ; Get address of code fragment
                                             ; to emulate 'dcmpg' instruction
           BX       R12                      ; Branch to code fragment.
                              |              ; Note use of BX rather than
                              |              ; BXJ because we do not want
                              |              ; this to be executed in HW
                              V
           Branch to dcmpg emulation code
                              |
                              V

           LDRB     R4, [R14, #1]!           ; Load next Java bytecode
                                             ; and update bytecode pointer
           FLDD     D1, [Rstack, #-8]!       ; Pop first operand from stack
                                             ; 1 Double = 2 stack words
           FLDD     D0, [Rstack, #-8]!       ; Pop second operand from stack
                                             ; 1 Double = 2 stack words
           LDR      R12, [Rexc, R4, LSL #2]  ; Get address of code fragment
                                             ; for next bytecode
           FCMPD    D0, D1                   ; Compare the 2 doubles
           FMSTAT                            ; Read result of compare
           MVNMI    R0, #0                   ; Result = -1 if <
           MOVEQ    R0, #0                   ; Result = 0 if =
           MOVGT    R0, #1                   ; Result = 1 if >
           STR      R0, [Rstack], #4         ; Push result to stack
           BXJ      R12                      ; Do next bytecode in
                                             ; hardware/software
```

FIG. 25

10081215.022502

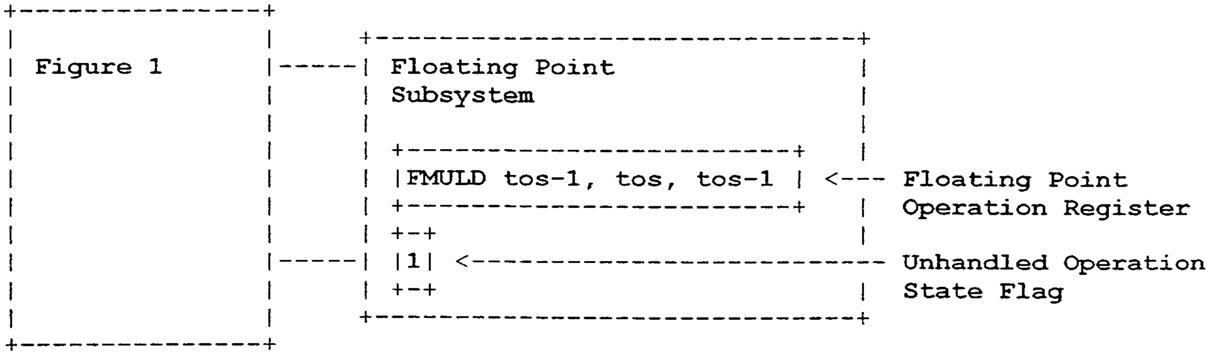
Figure 1
Floating Point
Subsystem
FMULD tos-1, tos, tos-1
1
Floating Point
Operation Register
Unhandled Operation
State Flag

FIG. 26

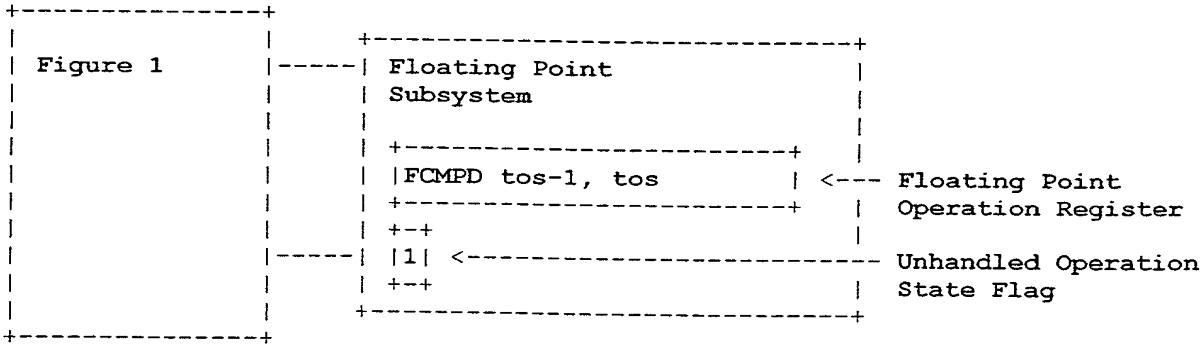
Figure 1
Floating Point
Subsystem
FCMPD tos-1, tos
1
Floating Point
Operation Register
Unhandled Operation
State Flag

FIG. 28

```
FMULD   D1, D2, D1              ; Executes as normal
FCMPD   D0, D1                  ; Causes unhandled operation
FMSTAT                          ; Unhandled operation signalled here
        |
        +----------------------+
         To ARM state to      |         ; unhandled operation handler executes
         emulate byte code    |         ; VFP read status instruction to
                        FMRX Rd, FPSCR  ; trigger VFP exception handling
                              |         ; if unhandled operation state flag
                              |         ; is set
                              +------------------------+
                 Unhandled operation state flag is     |
                 set so perform VFP exception handling |
                                                       |
                                     <Read Floating Point Operation Register>
                                     <Emulate FCMPD tos-1, tos>
                                     <Clear Unhandled Operation State Flag>
                                                       |
                              +------------------------+
                              | Return to unhandled operation handler
                <Re-attempt execution of FMRX Rd, FPSCR instruction,
                without triggering VFP exception this time>

                <Flush Java stack to memory>

                LDRB    R4, [R14]               ; Load bytecode which triggered
                                                ; unhandled operation
                LDR     R12, [Rexc, R4, LSL #2] ; Get address of code fragment
                                                ; to emulate 'dcmpg' instruction
                BX      R12                     ; Branch to code fragment.
                              |                 ; Note use of BX rather than
                              |                 ; BXJ because we do not want
                              |                 ; this to be executed in HW
                              V
                Branch to dcmpg emulation code
                              |
                              V

                LDRB    R4, [R14, #1]!          ; Load next Java bytecode
                                                ; and update bytecode pointer
                FLDD    D1, [Rstack, #-8]!      ; Pop first operand from stack
                                                ; 1 Double = 2 stack words
                FLDD    D0, [Rstack, #-8]!      ; Pop second operand from stack
                                                ; 1 Double = 2 stack words
                LDR     R12, [Rexc, R4, LSL #2] ; Get address of code fragment
                                                ; for next bytecode
                FMULD   D0, D0, D1              ; Retry the multiply, This will
                                                ; encounter the same problem as
                                                ; before, detected precisely,
                                                ; but executing ARM instructions
                                                ; rather than bytecodes. The VFP
                                                ; exception handler can handle
                                                ; this without problems
                FSTD    D0, [Rstack], #8        ; Push result, 2 words
                BXJ     R12                     ; Do next bytecode (dcmpg) in
                                                ; hardware/software
```

FIG. 27

10081215.022502

10081215.022502

```
FMULD  D1, D2, D1                ; Causes and signals unhandled operation
       |
       +--------------------------+
   To ARM state to                |             ; unhandled operation handler executes
   emulate byte code              |             ; VFP read status instruction to
                           FMRX Rd, FPSCR       ; trigger VFP exception handling
                                  |             ; if unhandled operation state flag
                                  |             ; is set. Since with precise unhandled
                                  |             ; operation detectection there is no
                                  |             ; unhandled operation state flag the
                                  |             ; VFP exception handling will never
                                  |             ; be triggered
                 <Flush Java stack to memory>

             LDRB    R4, [R14]                        ; Load bytecode which triggered
                                                      ; unhandled operation
             LDR     R12, [Rexc, R4, LSL #2]          ; Get address of code fragment
                                                      ; to emulate 'dmul' instruction
             BX      R12                              ; Branch to code fragment.
                      |                               ; Note use of BX rather than
                      |                               ; BXJ because we do not want
                      |                               ; this to be executed in HW
                      V
              Branch to dmul emulation code
                      |
                      V

             LDRB    R4, [R14, #1]!                   ; Load next Java bytecode
                                                      ; and update bytecode pointer
             FLDD    D1, [Rstack, #-8]!               ; Pop first operand from stack
                                                      ; 1 Double = 2 stack words
             FLDD    D0, [Rstack, #-8]!               ; Pop second operand from stack
                                                      ; 1 Double = 2 stack words
             LDR     R12, [Rexc, R4, LSL #2]          ; Get address of code fragment
                                                      ; for next bytecode (dcmpg)
             FMULD   D0, D0, D1                       ; Retry the multiply. This will
                                                      ; encounter the same problem as
                                                      ; before, detected precisely,
                                                      ; but executing ARM instructions
                                                      ; rather than bytecodes. The VFP
                                                      ; exception handler can handle
                                                      ; this without problems.
             FSTD    D0, [Rstack], #8                 ; Push result, 2 words
             BXJ     R12                              ; Do next bytecode (dcmpg) in
                                                      ; hardware/software
```

FIG. 29

10081215 .022502

```
FMULD  D1, D2, D1
FCMPD  D0, D1                       ; Causes and signals unhandled operation
       |
       +--------------------------+
   To ARM state to                |              ; unhandled operation handler executes
   emulate byte code              |              ; VFP read status instruction to
                            FMRX Rd, FPSCR       ; trigger VFP exception handling
                                  |              ; if unhandled operation state flag
                                  |              ; is set. Since with precise unhandled
                                  |              ; operation detectection there is no
                                  |              ; unhandled operation state flag the
                                  |              ; VFP exception handling will never
                                  |              ; be triggered
      <Flush Java stack to memory>

              LDRB    R4, [R14]                  ; Load bytecode which triggered
                                                 ; unhandled operation
              LDR     R12, [Rexc, R4, LSL #2]    ; Get address of code fragment
                                                 ; to emulate 'dcmpg' instruction
              BX      R12                        ; Branch to code fragment.
                          |                      ; Note use of BX rather than
                          |                      ; BXJ because we do not want
                          |                      ; this to be executed in HW
                          V
      Branch to dcmpg emulation code
                          |
                          V

              LDRB    R4, [R14, #1]!             ; Load next Java bytecode
                                                 ; and update bytecode pointer
              FLDD    D1, [Rstack, #-8]!         ; Pop first operand from stack
                                                 ; 1 Double = 2 stack words
              FLDD    D0, [Rstack, #-8]!         ; Pop second operand from stack
                                                 ; 1 Double = 2 stack words
              LDR     R12, [Rexc, R4, LSL #2]    ; Get address of code fragment
                                                 ; for next bytecode
              FCMPD   D0, D1                     ; Retry the comparison. This will
                                                 ; encounter the same problem as
                                                 ; before, detected precisely,
                                                 ; but executing ARM instructions
                                                 ; rather than bytecodes. The VFP
                                                 ; exception handler can handle
                                                 ; this without problems
              FMSTAT                             ; Read result of compare
              MVNMI   R0, #0                     ; Result = -1 if <
              MOVEQ   R0, #0                     ; Result = 0 if =
              MOVGT   R0, #1                     ; Result = 1 if >
              STR     R0, [Rstack], #4           ; Push result to stack
              BXJ     R12                        ; Do next bytecode in
                                                 ; hardware/software
```

FIG. 30